Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1997 Proceedings

Americas Conference on Information Systems (AMCIS)

8-15-1997

Software Artifact Reuse: A Domain Engineering Approach

Karma Sherif
Texas A&M University, karma@tamu.edu

Software Artifact Reuse: A Domain Engineering Approach

Karma Sherif
Department of Business Analysis and Research
Texas A&M University
College Station, Texas 77843-4217
karma@tamu.edu

Introduction

Reuse of software artifacts promises to reduce the cost and time of software development. However, most reuse efforts have been largely *opportunistic* in nature "based on convenience rather than long term design integrity" (Coglianese and Smith 1992). Academics and practitioners believe in the importance of building standardized reusable components that take into consideration the requirements specification of different applications. Domain Analysis and design were introduced for just this purpose.

Domain Engineering, which include Domain Analysis, Domain Design and Domain Implementation, focuses on the development of a whole family of systems by aggregating different solutions within the domain into a general one and packaging the general design for reuse.

This paper uses the ***Theory of Successive Abstractions*** to support the hypothesis that Domain Engineering increases the level of reuse. The theory states that memory encodes a general representation of things and uses them in representing information of different events. Likewise, Domain Engineering takes advantage of the similarities shared by systems in the domain and creates high level design that software developers can use as a reusable structure.

Domain Engineering

Domain Engineering constitutes three main phases:

1. Domain Analysis
2. Domain Design and
3. Domain Implementation.

Domain Analysis requires collecting experiences about a domain by acquiring information from the domain expert or through reverse engineering of existing systems. The result of the Domain Analysis stage is the **Domain Model**. A Domain Model provides a representation of the requirements of the domain.

Domain Design involves modeling the architecture and functional decomposition of the domain. This stage employs knowledge of past similar solutions to develop a new solution to recurring problems in a specific field. It identifies the domain constraints and boundaries and define the solution space. The deliverable is a **Domain Architecture**. It is a generic design for software systems in a domain.

Domain Implementation refers to the actual development of reusable assets. Domain Engineers use the Domain Architecture to build reusable components. The components include source code, test cases, documentation and CASE generated components.

The Theory of Successive Abstractions

The theory states that memory encodes a general abstract view of how things happen in some categories of cases and that "Structure is general enough to be of use in representing information from distinctly different events that are similar to the extent that they can share elements of the same structure." (Schank 82) This entails removing the details and abstracting contents of the components in memory to a degree where they can be applied to different situations within a particular domain.

Software repositories like human memory should not be crowded by every individual experience but rather organized into abstract representation of similar cases. Domain engineering works to identify "reusable requirements specification, and reusable architecture" for families of systems that constitute different application domains (Gomaa & Kerschberg 1995). The goal is to synthesize the similarities shared by systems in the domain into a high level designs that software developers can use as a structure into which reusable components can be plugged.

Conclusion

Software reuse will not achieve its expected benefits unless reuse is systematized. Components should be built specifically for reuse. Domain Engineering takes advantage of the similarities inherent in systems within one domain and build a domain model that defines the basic structure for existing and future requirements. This paper draws on the theory of successive abstractions as a theoretical construct for supporting the hypothesis that Domain Engineering increases reuse levels.

References available upon request from author